SyScan 360

BROWSER FUZZING IN 2014:

DAVID VS GOLIATH

aka

Learn where to throw your stones

ROSARIO VALOTTA

AGENDA

1. MEMORY CORRUPTION BUGS: THE WHERE AND THE WHYS
2. BROWSER FUZZING: THE STATE OF THE ART
3. INTRODUCTION OF A NEW FUZZING APPROACH
4. FUZZING WITH TIME
5. FUZZING WITH SPACE
6. ENJOY FILEJA!
7. SOME RESULTS

你好！

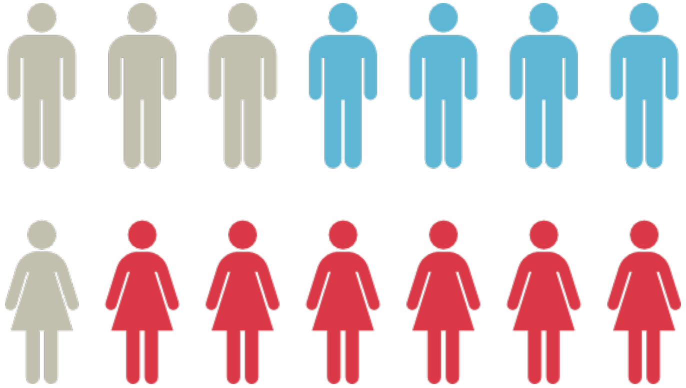
- HI BEIJING! I AM ROSARIO AND I COME FROM 8.126,23 KMs AWAY…
- DAILY JOB: PROJECT MANAGER IN MOBILE TELCO OPERATOR
- INDIPENDENT SECURITY RESEARCHER FOR FUN AND PASSION (AND NO MONEY…)
- MAINLY FOCUSED ON WEB SECURITY, BROWSER SECURITY AND NEW ATTACK TECHNIQUES
- SPEAKER AT SEC CONFERENCES:
  - HITB (X2) – DEEPSEC – NUIT DU HACK – PHDAYS – SWISS CYBER STORM
- HTTPS://SITES.GOOGLE.COM/SITE/TENTACOLOVIOLA/

WHY FUZZING?

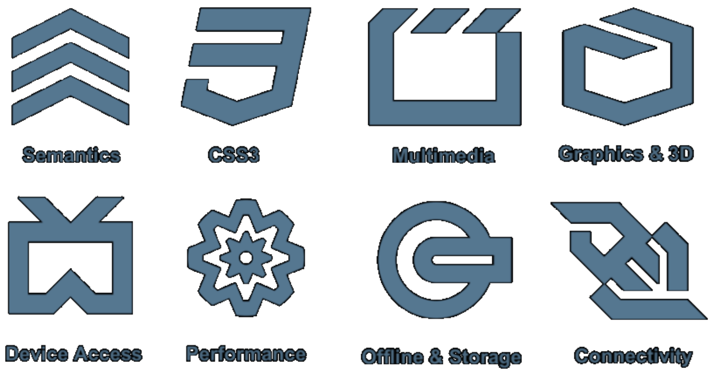
QUICK ANSWER: SPOTTING MEMORY CORRUPTION VULNERABILITIES

…AND WHY BROWSERS?

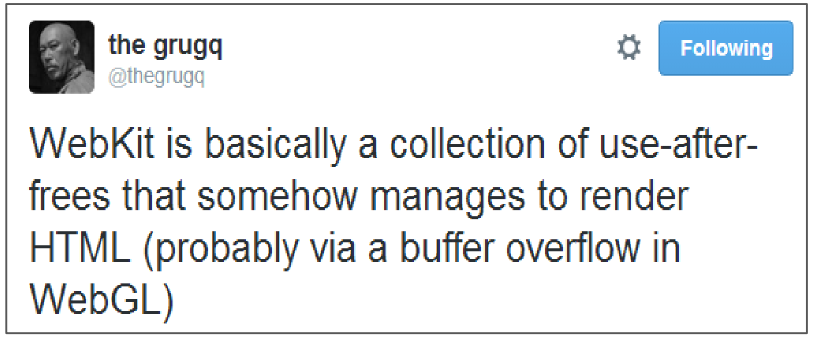
UBIQUITOUS PLATFORM: USED BY BILLIONS USERS EVERYDAY



HUGE ATTACK SURFACE: NEW FEATURES ADDED ON EVERY RELEASE, MILLIONS LINES OF CODE



COMPLEXITY COMES AT A PRICE



the grugq
@thegrugq

WebKit is basically a collection of use-after-frees that somehow manages to render HTML (probably via a buffer overflow in WebGL)

VULNERABILITIES MARKET

PROVIDERS

INDIPENDENT RESEARCHERS

VULNERABILITIES RESEARCHING COMPANIES

BROWSER VENDORS

BUYERS/BROKERS

BROWSERS BOUNTY PROGRAMS

$

VULNERABILITIES BROKERS COMPANIES

$ $

SPYING BUSINESS

$ $ $

BAD GUYS

$ $ $ $

FINAL CUSTOMERS

GOVERNMENTS

LAW ENFORCEMENT AGENCIES

MILITARY ORGANIZATIONS

BIG CORPORATIONS

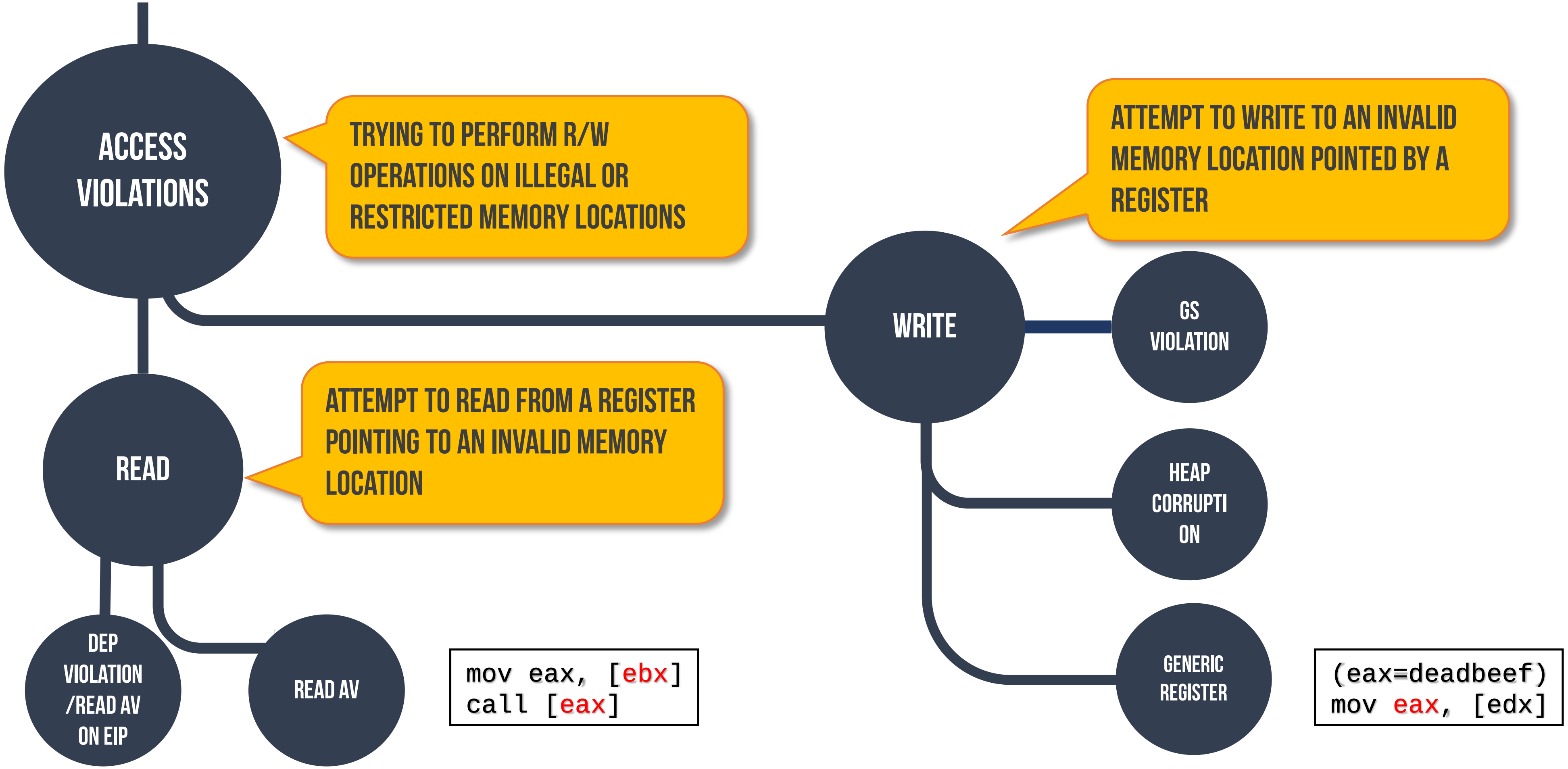
ANYONE WITH ENOUGH BUDGET AND WILLING TO CONTROL ITS TARGETS

DAVID VS GOLIATH

- FINDING EXPLOITABLE VULNERABILITIES IS BECOMING INCREASINGLY HARDER
- BROWSERS ARE BECOMING MORE SECURE
- BIGGER AND COMPETITIVE MARKET
- IF YOU'RE A LONELY SECURITY RESEARCHER WITH A SLINGSHOT YOU CANNOT COMPETE WITH THAT BUG-KILLING ARMADA OUT THERE…
- OLD APPROACHES DON'T WORK ANY MORE
- YOU NEED NEW IDEAS AND A NEW APPROACH, YOU NEED TO KNOW WHERE TO THROW YOUR STONES.

MEMORY CORRUPTION VULNS: A PRIMER

- SPECIAL KIND OF ACCESS VIOLATION BUGS
- AV BUG IS SAID TO EXPLOITABLE IF:
  - IT ALLOWS TO CONTROL EXECUTION FLOW
  - THE ATTACKER HAS FULL CONTROL ON THE REGISTER CAUSING THE VIOLATION



TRYING TO PERFORM R/W OPERATIONS ON ILLEGAL OR RESTRICTED MEMORY LOCATIONS

ATTEMPT TO WRITE TO AN INVALID MEMORY LOCATION POINTED BY A REGISTER

ATTEMPT TO READ FROM A REGISTER POINTING TO AN INVALID MEMORY LOCATION

```
mov eax, [ebx]
call [eax]
```

```
(eax=deadbeef)
mov eax, [edx]
```

MEMORY PROTECTIONS AND SANDBOXES

IN MODERN BROWSERS AND OSs CONTROLLING EIP IS NOT ENOUGH TO GAIN ARBITRARY CODE EXECUTION

OS MEMORY PROTECTIONS

DATA EXECUTION PREVENTION: NOT EXECUTABLE STACK, MUST CHAIN ROP GADGETS TO COMPOSE AN EXECUTABLE SHELLCODE

ASRL: HIGH ENTROPY RANDOMIZATION, CANNOT RELY ON FIXED MEMORY ADDRESSES WHEN JUMPING TO ROP GADGETS

/GS: STACK BUFFER SECURITY CHECK, CANNOT OVERFLOW STACK USING ATTACKER INPUT

SAFE SEH: EH SECURITY CHECK, CANNOT OVERWRITE EH ROUTINE TO GET CODE EXECUTION ON EXCEPTIONS

BROWSER PROTECTIONS

SANDBOX/PROTECTED MODE: EVERY BROWSING WINDOW RUNS WITH LOW PRIVILEGES AND CANNOT ACCESS TO FILE SYSTEM OR OTHER CRITICAL MACHINE’S DATA STRUCTURES

SAFE BROWSING/SMARTSCREEN: TECHNOLOGIES TO BLOCK EXECUTION OF DOWLOADED FILES BASED ON FILE SIGNATURE AND REPUTATION

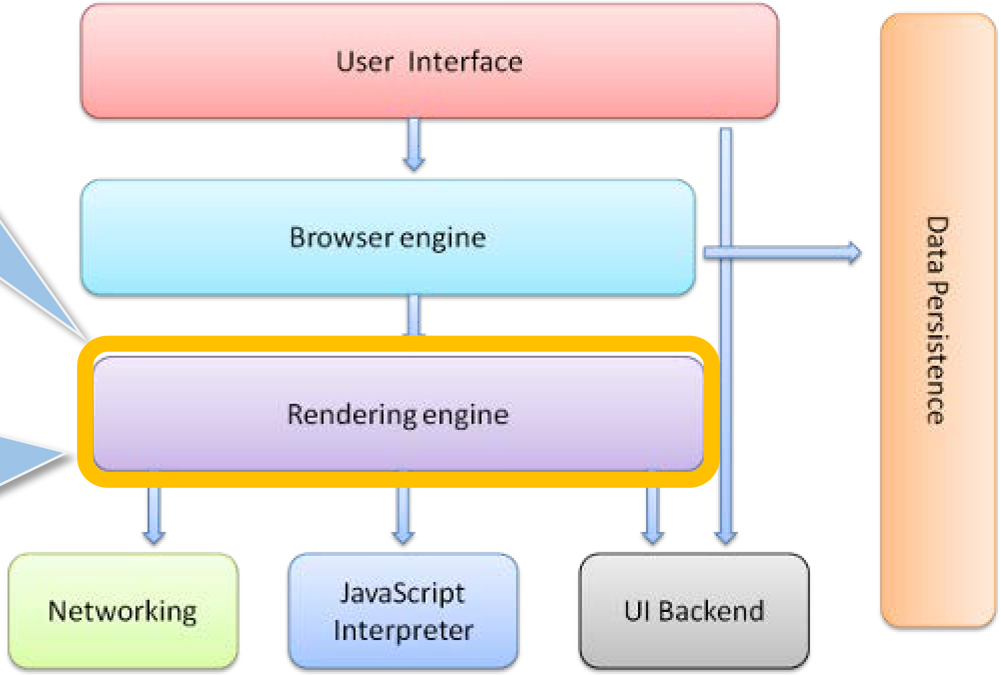
EVEN IF SOME BYPASS TECHNIQUES EXIST, DEFEATING MEMORY PROTECTIONS AND EVADING BROWSER SANDBOXES CAN BE A TOUGH WORK.

USUAL TARGETS FOR BROWSER FUZZING

RENDERING ENGINE IS THE MOST COMPLEX MODULE OF BROWSER ARCHITECTURE: DISPLAYS HTML ,XML, SVG, MATHML, VML DOCUMENTS AND IMAGES.
IT CAN DISPLAY OTHER TYPES OF DATA VIA PLUG-INS OR EXTENSIONS (PDF, MEDIA FILE, FONTS, ETC)

- IT IS ITS RESPONSIBILITY TO PARSE HTML, APPLY CSS AND BUILD AN INTERNAL TREE MODEL OF THE WEB PAGE CALLED “DOM”

- EVERY LOGICAL OPERATION PERFORMED ON THE WEB TREE IS EXECUTED ON THE DOM BEFORE RENDERING IS DONE



WEAPONS OF CHOICE TO EFFECTIVELY FUZZ RENDERING ENGINE ARE:

1. FUZZING FILE FORMATS
2. FUZZING DOM

<FILE/>FORMAT//FUZZING

DUMB MUTATION FUZZING

START WITH A POOL OF VALID TESTCASES

PERFORM PSEUDO-RANDOM MUTATIONS ON THEM (BIT FLIPPING, APPEND RANDOM,ETC)

SEVERAL FREE TOOLS AVAILABLE:

- ZZUF
- RADAMSA
- SDL MINIFUZZ

+ EASIER APPROACH

- LESSER CODE COVERAGE AND CODE PATHS

+ MAY PROVIDE UNEXCPECTED RESULTS EVEN IN THE LONG RUN

GENERATION FUZZING

MODEL YOUR INPUT DEFINING STRUCTURE OF DATA BLOCKS (GRAMMAR AWARE FUZZING)

DEFINE A STATE MODEL AND AGENTS (OPERATION AVAILABLE ON DATA)

SEVERAL VALUABLE TOOLS:

- PEACH
- SPIKE

- REQUIRE DEEP KNOWLEDGE OF PROTOCOL/FILE FORMAT

+ BETTER COVERAGE, BECAUSE OF VALID COMBINATIONS

+ COMPREHENSIVE AND MORE SCALABLE

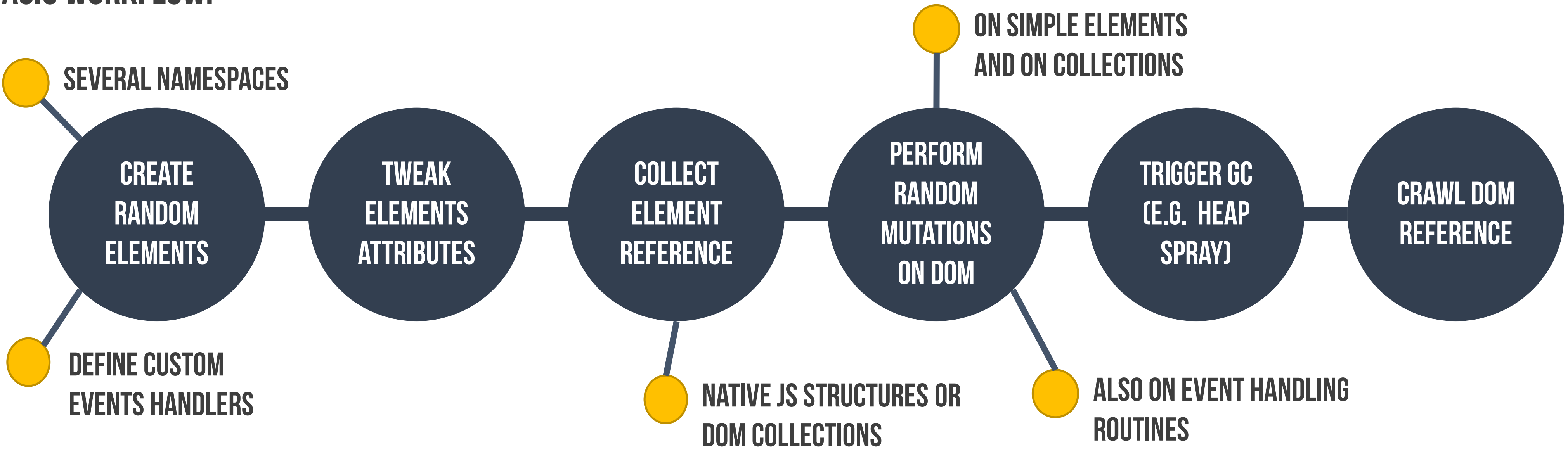
DOM FUZZING

- DOM HAS A COMPLEX DATA STRUCTURE, THAT CARRIES THE HEAVY LOAD OF MODELING ALL POSSIBLE OPERATIONS ENABLED BY MODERN HTML – JS – CSS
- LOT OF APIs: DOM SPECIFICATIONS ARE DEFINED BY W3C AND DIVIDED IN 4 DIFFERENT DOCUMENTS

- CROSSFUZZ BY MICHAL ZALEWSKI IS STILL THE BENCHMARK
  - LOT OF MODS, WIDESPREAD COVERAGE → HARD TO SPOT NEW CRASHES

- NDUJA INTRODUCED SOME NEW CONCEPTS FROM DOM LEVEL 2 AND 3 SPECS AND PROVIDED INTERESTING RESULTS

- BASIC WORKFLOW:



CREATE RANDOM ELEMENTS

TWEAK ELEMENTS ATTRIBUTES

COLLECT ELEMENT REFERENCE

PERFORM RANDOM MUTATIONS ON DOM

TRIGGER GC (E.G. HEAP SPRAY)

CRAWL DOM REFERENCE

SEVERAL NAMESPACES

DEFINE CUSTOM EVENTS HANDLERS

NATIVE JS STRUCTURES OR DOM COLLECTIONS

ON SIMPLE ELEMENTS AND ON COLLECTIONS

ALSO ON EVENT HANDLING ROUTINES

A NEW IDEA: FUZZING DOM IN TIME & SPACE

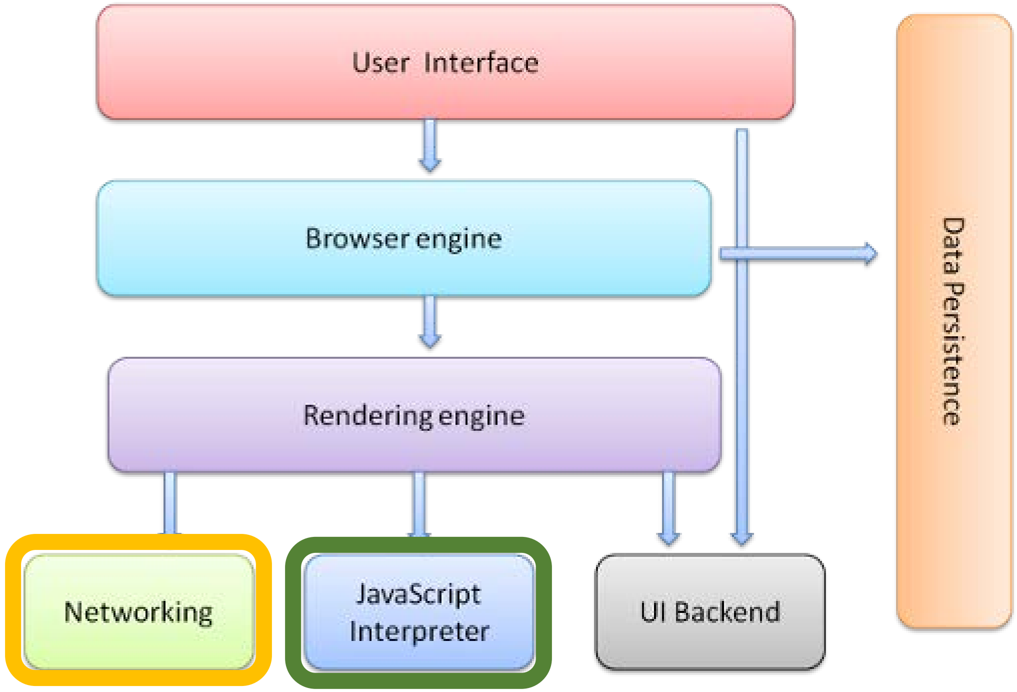
- LET'S PROPOSE AN EXTENDED SURFACE FOR BROWSER DOM FUZZING BEYOND RENDERING ENGINE + FILE FORMAT PARADIGM

- THE IDEA IS TO INTRODUCE TWO NEW DIMENSIONS TO THE FUZZING MODEL: TIME AND SPACE

- TIME: INTRODUCING TIME DEPENDENCIES INTO YOUR FUZZING LOGIC:
  1. SYNCH / ASYNCH EVENTS
  2. NETWORK INTERACTIONS

  IN ORDER TO TRIGGER RACE CONDITIONS

- SPACE: EXTEND YOUR FUZZING LOGIC PERIMETER IN ORDER TO FIND MEMORY INCONSISTENCIES ACROSS MULTIPLE SCRIPTING CONTEXTS



STONE #1 - FUZZING/WITH/TIME

JS RACE CONDITIONS

- ALL MODERN BROWSERS IMPLEMENT THEIR JS ENGINES USING ONE OS-THREAD.
- THE ONLY EXCEPTION ARE WEB WORKERS, BUT WITH LITTLE SECURITY RISK AS THEY DON'T ACCESS DOM.

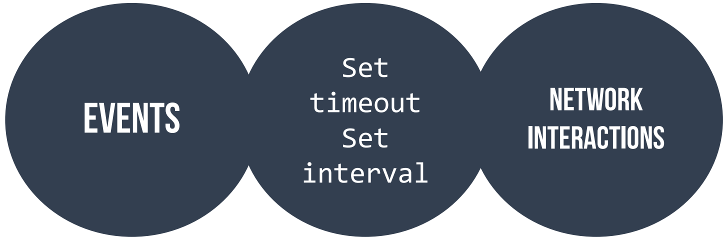
QUESTION: GIVEN THAT 2 JS EVENTS CANNOT HAPPEN AT THE SAME TIME, DO I REALLY NEED TO CARE ABOUT RACE CONDITIONS?

- SHORT ANSER: YES
- LONG ANSWER: YOU MAY BE IN TROUBLE IF AN OBJECT/FUNCTION YOUR CODE RELIES ON IS CHANGED BETWEEN WHEN AN EVENT IS FIRED AND THE CALLBACK IS CALLED

- SEVERAL RACE CONDITION VULNS HAVE BEEN SPOTTED IN THE PAST:
  - APPLE WEBKIT - CVE-2012-3748
  - MOZILLA FIREFOX - CVE-2006-4253
  - GOOGLE CHROME - CVE-2006-4253
  - MICROSOFT IE - CVE-2011-1257

  BUT NO TARGETED FUZZING ALGORITHM TO STRESS RACE CONDITION INSURGENCE

- THREE MAIN SOURCES OF TROUBLE:



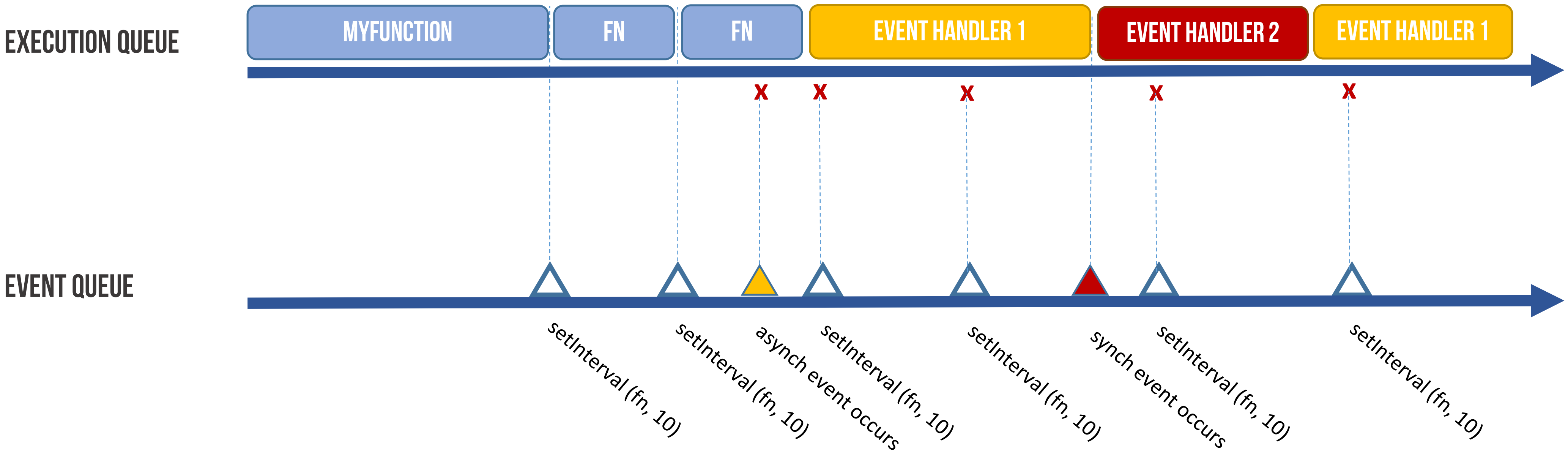
EVENTS

Set timeout Set interval

NETWORK INTERACTIONS

JS TIMING MODEL

- BROWSERS ARE EVENT-DRIVEN
- ALMOST EVERY ACTION PERFORMED ON A BROWSER RESULTS IN AN EVENT BEING GENERATED AND APPENDED TO THE EVENT QUEUE
- EVENT LOOP IS A FUNCTION THAT TAKES EVENTS FROM THE QUEUE AND PROCESS THEM WHEN TIME PERMITS → ONLY ONE RUNNING SCRIPT AT A TIME
- MOST EVENTS ARE PROCESSED ASYNCHRONOUSLY
- SOME SPECIAL EVENTS (`mutation`) AND EVENTS FIRED WITH `dispatchEvent` ARE SYNCHRONOUS

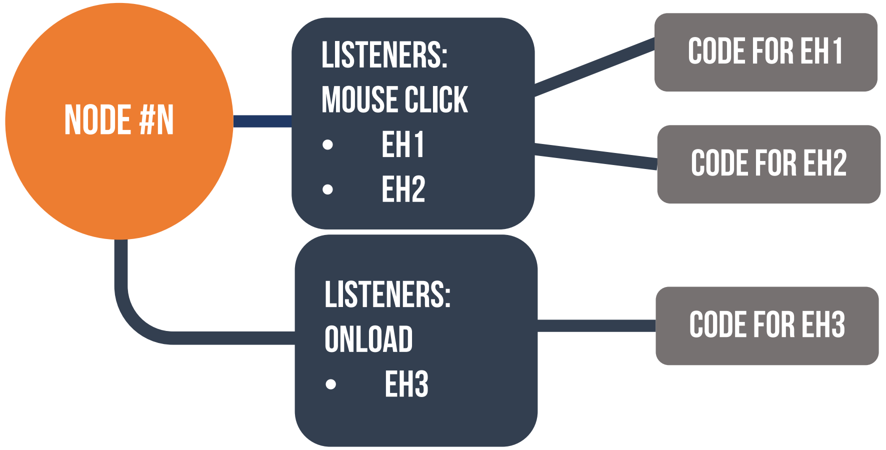


JS EVENT MODEL

- DOM LEVELS 3 AND 4 DEFINE HOW EVENTS ARE FIRED, HANDLED AND HOW TO MANAGE EVENT LISTENERS FOR YOUR DOM OBJECTS

```
•   myelem.addEventListener(“MouseClick”, myHandlerFunction, captureIsOn)
•   myelem.removeEventListener(“MouseClick”, myHandlerFunction, captureIsOn)
```

- DOM HOLDS A MAP OF LISTENERS AND EVENT HANDLERS CONNECTED TO EACH NODE KEYED BY EVENT TYPE



- EVENT TYPES:
  - UI EVENTS
  - MOUSE EVENTS
  - MUTATION EVENTS
  - ETC

```
•   DOMAttrModified
•   DOMAttributeNameChanged
•   DOMCharacterDataModified
•   DOMElementNameChanged
•   DOMNodeInserted
•   DOMNodeInsertedIntoDocument
•   DOMNodeRemoved
•   DOMNodeRemovedFromDocument
•   DOMSubtreeModified
```

SYNCH EVENTS

EVENTS PROPAGATION

- EVENT OBJECTS ARE DISPATCHED TO AN EVENT TARGET

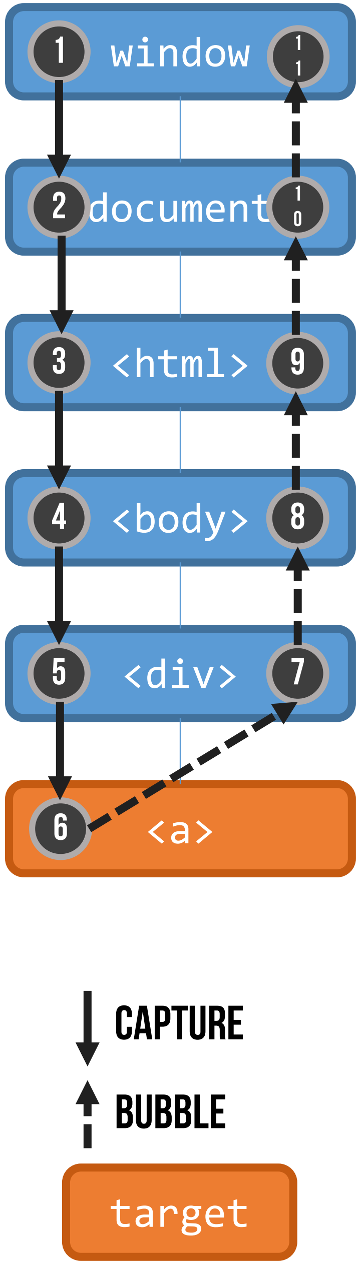
- AT THE BEGIN OF THE DISPACTH, BROWSER MUST DETERMINE THE PROPAGATION PATH FOR THE EVENT

  - A HIERARCHICAL LIST OF DOM NODES THROUGH WHICH THE EVENT MUST PASS

- PROPAGATION PATH IS DIVIDED IN 3 PHASES:
  1. CAPTURE
  2. TARGET
  3. BUBBLE

- YOU CAN CUSTOMIZE THE EVENT HANDLER ROUTINE ACCORDING TO THE `event.phase`

- YOU CANNOT ALTER THE PROPAGATION PATH AFTER THE EVENT HAS FIRED, EVEN IF SOME NODES ARE REMOVED OR DOM TREE IS MODIFIED MAKING THE PROPAGATION NON CONTINUABLE



LOOK MA' MORE EVENTS!

- STARTING WITH DOM LEVEL 4, `MutationEvents` ARE DEPRECATED FOR PERFORMANCE REASONS

- THIS DOES'NT MEAN YOU CANNOT USE THEM ANYWAY ;-)

- ALL BROWSERS NOW SUPPORT `MutationObservers`

- EVERY MUTATION IS NOW QUEUED IN A `mutation` COLLECTION

- THIS MAKE DOM MUTATION EVENTS **ASYNCHRONOUS**

```
// select the target node
var target = document.querySelector('#some-id');
// create an observer instance
var observer = new MutationObserver(function(mutations) {
         mutations.forEach(function(mutation) {
                  //whatever
         });
});
// configuration of the observer:
var config = { attributes: true, childList: true};
// pass in the target node and the observer options
observer.observe(target, config);
// later, you can stop observing
observer.disconnect();
```

- SOME INTERESTING APIs WILL PROVIDE AN ARRAY/LIST OF MUTATED NODES, WITH A CHANCE TO DISTINGUISH ADDED NODES AND REMOVED ONES

- `mutations = observer.takeRecords();`
- `NodeList nl=observer.addedNodes;`
- `NodeList nl=observer.removedNodes;`

FUZZING WITH EVENTS

⚠ `setTimeout(fn,0)` WILL FORCE `fn` TO BE EXECUTED IN THE NEXT TICK EVEN IF NOT THE FIRST EVENT IN QUEUE

TRY TO MODIFY `delay` WHILE `fn` IS ALREADY ON THE EVENT QUEUE

⚠ SYNCH EVENTS CAN BE NESTED

A MUTATION EVENT HANDLER CAN BE SYNCHRONOUSLY INTERRUPTED BY AN OTHER MUTATION EVENT TRIGGERED IN THE HANDLER CODE (E.G. A DOM NODE IS REMOVED OR SOME OTHER EVENT IS DISPATCHED USING `dispatchEvent`)

⚠ PROPAGATION PATH REMAINS IMMUTABLE AFTER DISPATCHING

UNPREDICTABLE CONSEQUENCES MAY COME FROM ALTERING NODES BELONGING TO PROPAGATION PATH DURING CAPTURE OR BUBBLE PHASES

⚠ `takeRecords`, `addedNodes`, `removedNodes` NOTHING BUT A LOGICAL VIEW (COLLECTION OF REFERENCES) OF MUTATED RECORDS OF DOM NODES

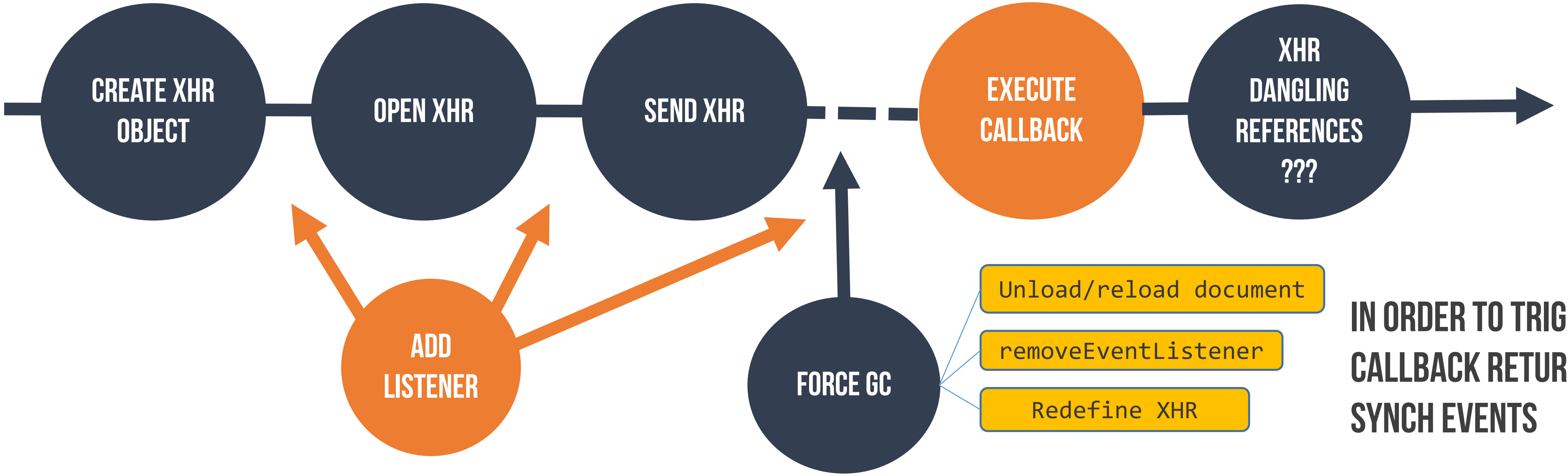
LISTEN TO MUTATION EVENTS ON A GIVEN DOM NODE USING BOTH `MutationEvent` BOTH `MutationObservers`: TRY TO TWEAK `takeRecords` ARRAY AFTER SOME SYNCH `MutationEvent` ALTERED THE NODE

FUZZING WITH XHR (1/2)

SIMILAR CONSIDERATIONS ALSO APPLY TO WEB SOCKETS NETWORK CALLS

- A LOT MORE RACE CONDITIONS MAY BE TRIGGERED USING `XMLHttpRequests`
- XHRs CAN BE SYNCH (DEPRECATED) OR ASYNCH OPERATIONS
- XHRs STATE CAN BE MONITORED USING SOME EVENT LISTENERS:
  - `readystatechange, progress, abort, error, load, timeout, loadend`

- AN XHR OBJECT MUST NOT BE GARBAGE COLLECTED IF ITS STATE IS OPENED AND THE SEND FLAG IS SET OR ITS STATE IS LOADING OR IT HAS ONE OR MORE EVENT LISTENERS REGISTERED

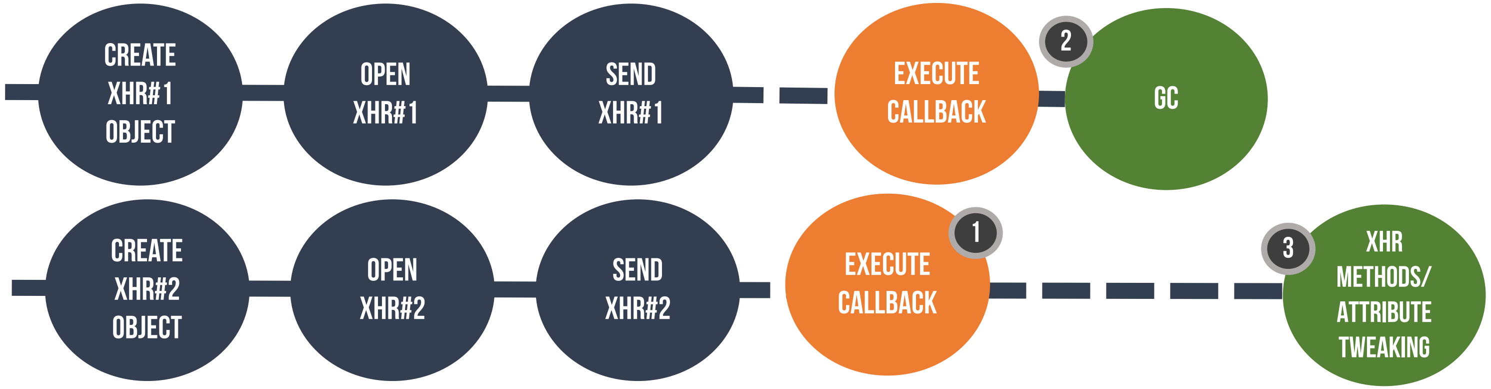


IN ORDER TO TRIGGER GC BEFORE CALLBACK RETURNS YOU CAN RELY ON SYNCH EVENTS

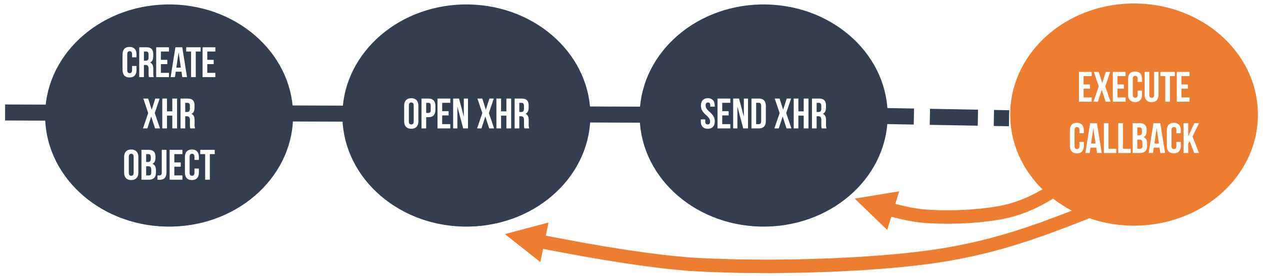
FUZZING WITH XHR (2/2)

SIMILAR CONSIDERATIONS ALSO APPLY TO WEB SOCKETS NETWORK CALLS

- A RACE CONDITION MAY HAPPEN IF YOUR CALLBACK CODE RELIES ON SOME OBJECT/FUNCTION THAT HAS BEEN GC'ED OR IS UNINITIALIZED AT THE MOMENT OF CALLBACK EXECUTION
- E.G. SUPPOSE YOUR CALLBACK CODE EXECUTES SOME MUTATION OPERATIONS ON AN OBJECT BOUND TO XHR AND YOU'RE RUNNING MULTIPLE CONCURRENT XHR CALLS



- SOME OTHER RACE CONDITIONS MAY HAPPEN WHEN XHR ARE RECURSIVE:



LET'S NETWORK PARTY WITH DOM!

- THE IDEA HERE IS TO COMBINE CLASSICAL APPROACH OF DOM FUZZING WITH NETWORK CALLS
- SNIPPET OF VALID JS ARE RETRIEVED USING XHRs OR WSs AND PROCESSED IN CONTEXT OF THE DOM

MIX SYNCH AND ASYNCH CALLS

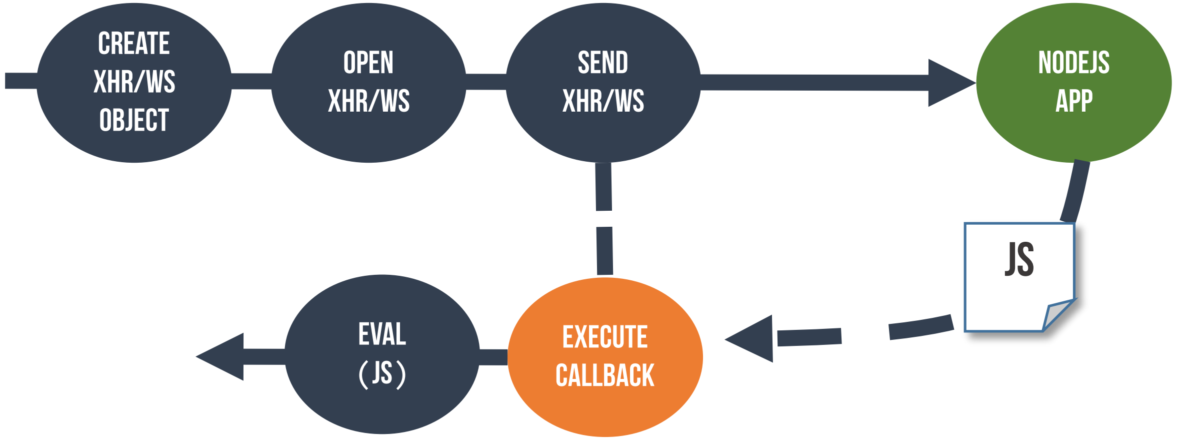
```
xhr = new XMLHttpRequest();
xhr.open("GET", "http://127.0.0.1:8887", RBool());
xhr.onreadystatechange = function() {
        if (this.readyState == 4) {
                var s=document.createElement("script");
                s.innerText=xhr.responseText;
                document.body.appendChild(s);
        }
}
```

```
socket = new WebSocket("ws://127.0.0.1:9999", "fuzz");
socket.addEventListener("message", function(event) {
        s=document.createElement("script");
        s.src=(window.URL.createObjectURL(new Blob([data],{type:"text/html"}));
        document.body.appendChild(s);
        f.contentWindow.eval(s.innerText);
});
```

...THE OTHER SIDE OF THE PARTY

- ON THE SERVER SIDE THERE ARE A BUNCH OF node JS APPLICATIONS, IMPLEMENTING HTTP AND WS SERVERS

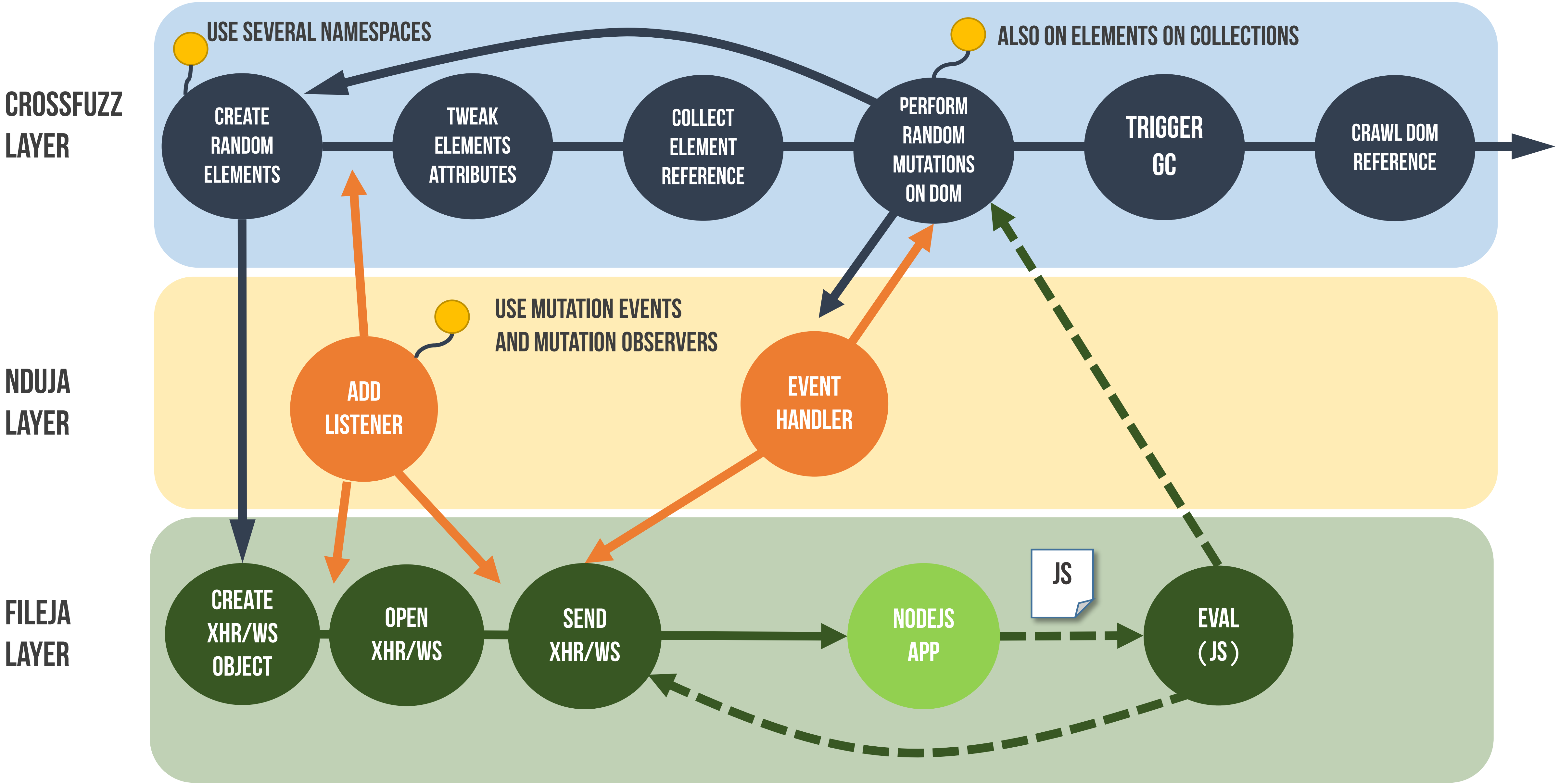
- FOR EVERY REQUEST:
  1. A RANDOM DELAY IS APPLIED BEFORE GENERATING THE RESPONSE → THIS AFFECT TIMING ON CLIENT SIDE
  2. A FRAGMENT OF VALID JS IS COMPOSED AND RETURNET AS `text/html` OR…
  3. …A REFERENCE TO A FUNCTION DECLARED ON THE CLIENT SIDE IS RETURNED



- FUZZING WITH CODE FRAGMENTS HAS BEEN AN APPROACH USED IN THE PAST BY LANGFUZZ, BUT HERE THE GOAL IS TO TARGET SPECIFIC BORDERLINE EXECUTION SCENARIOS → RACE CONDITIONS

- THIS EVALUATION OF THE JS FRAGMENT IS INFLUENCED BY:
  - SYNCH DOM MUTATIONS THAT OCCURRED IN THE MIDDLE OF CALL PROCESSING
  - XHR/WS REFERENCES NOT DISPOSED WHEN CLIENT LOCATION PAGE IS NAVIGATED AWAY
  - RACE CONDITIONS IN REQUEST/RESPONSE MANAGEMENT

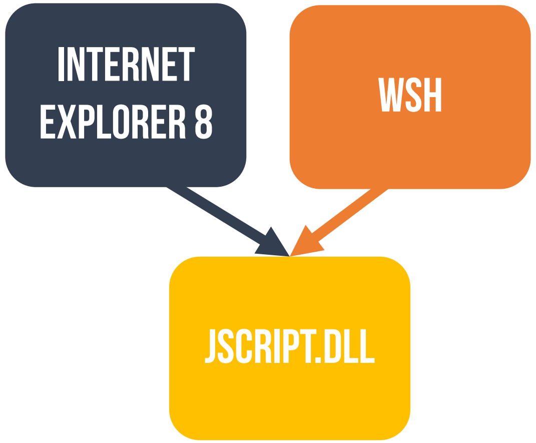
THE MASTERPLAN #1

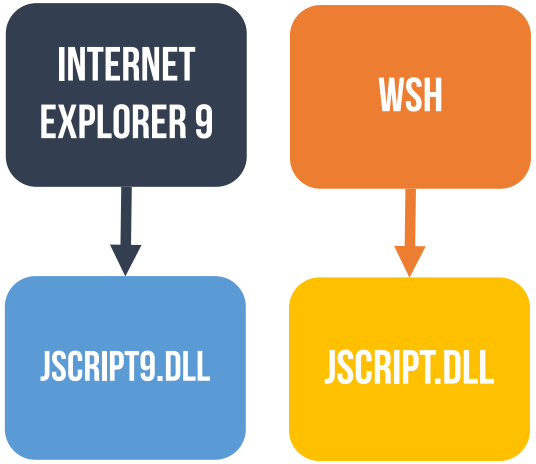


STONE #2 - FUZZING/WITH/SPACE

IE' SCRIPTING ENGINES

- IN WINDOWS, JAVASCRIPT IS IMPLEMENTED AS A COM DLL THAT CAN BE HOSTED BY VARIOUS APPLICATIONS:
    - WINDOWS SCRIPT HOST (WSH)
    - INTERNET EXPLORER
- BEFORE VERSION 9, IE USED THE SAME JAVASCRIPT ENGINE AS WSH - JSCRIPT.DLL
- IN IE9 A DIFFERENT DLL HAS BEEN SHIPPED – JSCRIPT9.DLL, DESIGNED SPECIFICALLY FOR THE BROWSER
- TO MAKE IT BETTER BACKWARD COMPATIBLE, JSCRIPT9 ENGINE CAN EMULATE IE8 AND IE7 DOC MODES, BY KEEPING THE SAME BEHAVIOR AS LEGACY ENGINE
- EMULATING THE LEGACY ENGINE DOESN'T ACTUALLY MEAN LOADING THE LEGACY ENGINE!





USING LEGACY ENGINES

- BY DEFAULT WHEN DEFINING `<script>` OR `<script language=‘javascript’>` IE 9+ LOADS JSCRIPT9.DLL

- YOU CAN FORCE IE TO LOAD LEGACY ENGINE BY DECLARING:
  `<script language=‘Jscript.Encode’>`

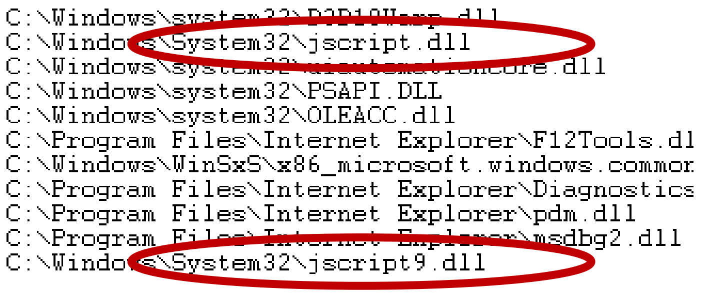
- `Jscript.Encode` WAS DESIGNED FOR INTERPRETING ENCODED SCRIPTS, BUT ALSO WORKS FOR CLEAR TEXT ONES AND IS NOT SUPPORTED IN JSCRIPT9

- IN ORDER TO BE ABLE TO LOAD `Jscript.Encode` SCRIPTS YOU HAVE TO FORCE IE=8 EMULATION

- IE ALSO SUPPORTS A VBSCRIPT SCRIPTING ENGINE MANAGED ALWAYS BY JSCRIPT.DLL
  `<script language=‘vbscript’>`

- VBSCRIPT IS DEPRECATED STARTING WITH IE11, BUT YOU CAN SUMMON IT BACK FORCING BROWSER TO WORK IN EMULATION MODE OF PREVIOUS VERSIONS

```
<html>
<head>
<title> test encode</title>
<meta http-equiv="X-UA-Compatible"
content="IE=8"></meta>
</head>
<body>
<script  language="Jscript.Encode">
    function a(){alert(1);}
</script>
<script> a();</script>
</body>
</html>
```



```
C:\Windows\system32\D3D10Warp.dll
C:\Windows\System32\jscript.dll
C:\Windows\system32\uiautomationcore.dll
C:\Windows\system32\PSAPI.DLL
C:\Windows\system32\OLEACC.dll
C:\Program Files\Internet Explorer\F12Tools.dl
C:\Windows\WinSxS\x86_microsoft.windows.commor
C:\Program Files\Internet Explorer\Diagnostics
C:\Program Files\Internet Explorer\pdm.dll
C:\Program Files\Internet Explorer\msdbg2.dll
C:\Windows\System32\jscript9.dll
```

HOST - ENGINE INTERACTIONS

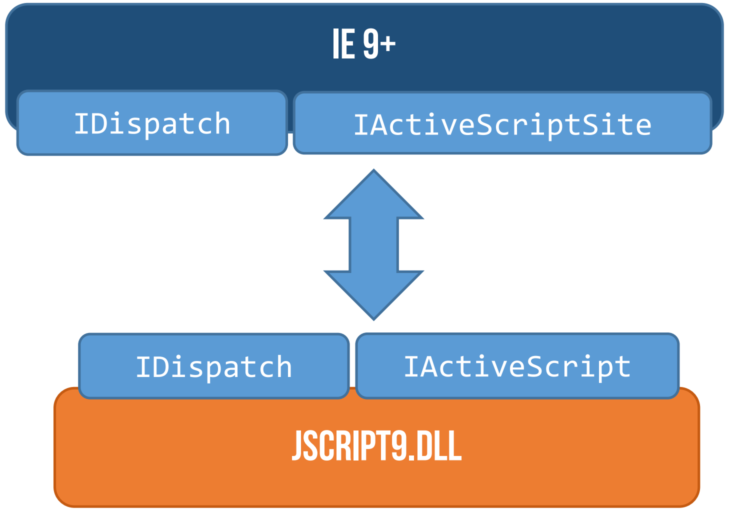
- COMMUNICATION BETWEEN THE HOST AND THE SCRIPT ENGINE IS CARRIED ON USING A PAIR OF INTERFACES : `IActiveScript` AND `IActiveScriptSite`

- `IActiveScriptSite` IS IMPLEMENTED ON THE HOST SIDE AND ENABLES THE JS ENGINE TO CALL ITS HOST

- `IActiveScript`, IMPLEMENTED ON THE SCRIPT ENGINE SIDE, PROVIDES NECESSARY FUNCTION CALLS TO INITIALIZE THE ENGINE

- `IActiveScript` ALSO INCLUDES THE API `AddNamedItem` USED BY THE HOST TO ADD OBJECTS TO ENGINE'S GLOBAL SCOPE. IE USES THIS FUNCTION TO ADD WINDOW AND OTHER DOM OBJECTS INTO THE ENGINE

- `Idispatch` INTERFACE IS IMPLEMENTED ON BOTH SIDES AND IS USED TO RETRIEVE HANDLES OF OBJECTS AND EXECUTE GET, SET AND FUNCTION CALLS OPERATIONS ON THEM

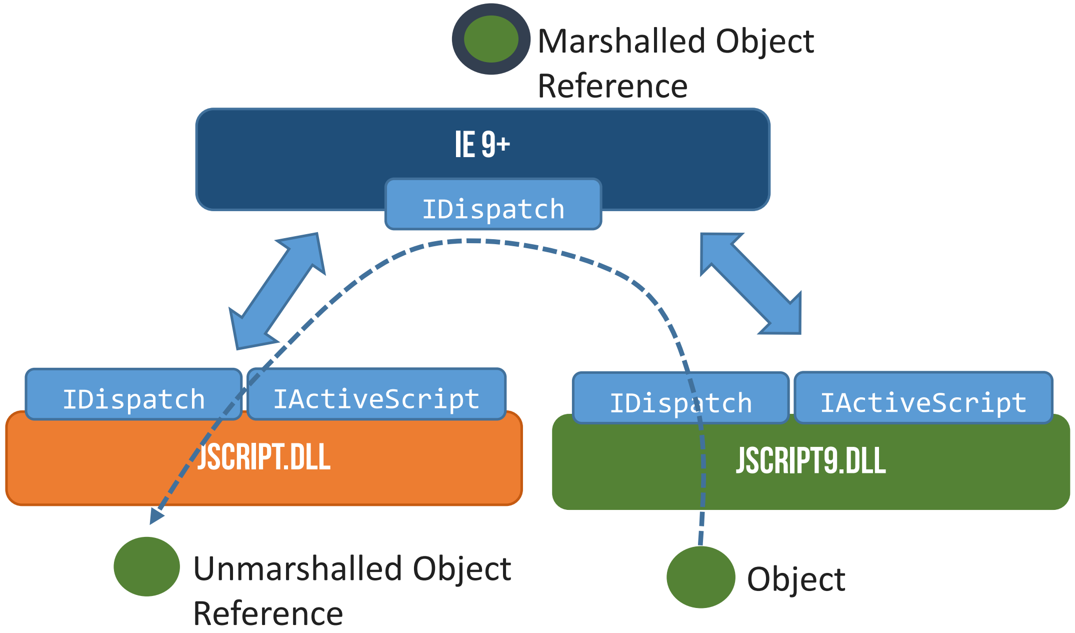


CROSS ENGINES INTERACTIONS

- WHEN A `Jscript.Encode` SCRIPT IS FOUND, IE9 HOSTS BOTH JSCRIPT9 AND JSCRIPT ENGINES AT RUNTIME AND BOTH ENGINES CAN TALK TO THE OTHER ONE

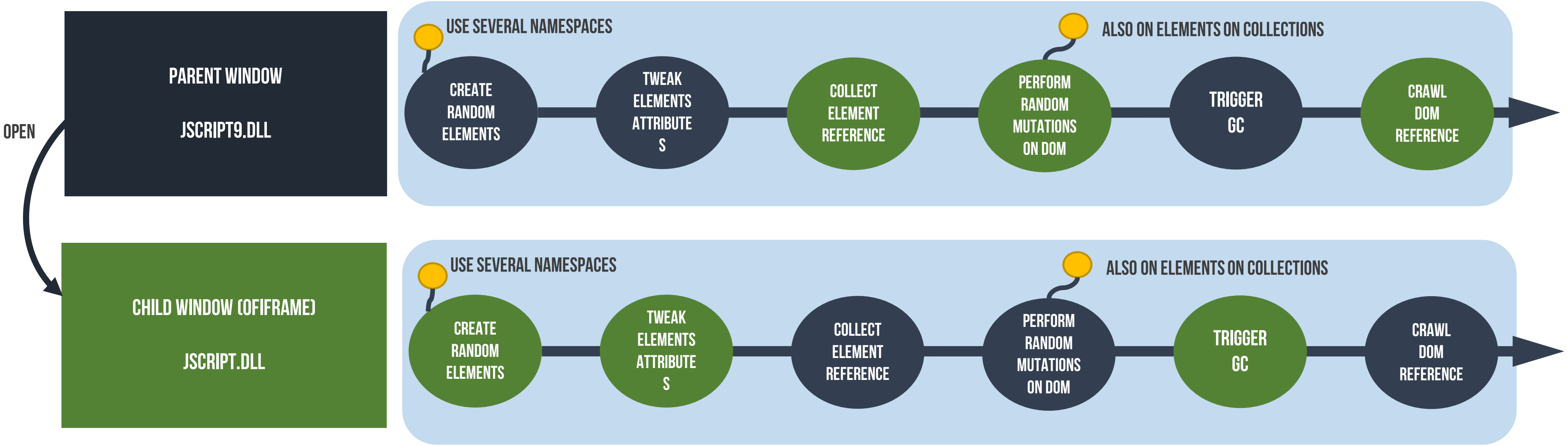
- WHEN AN ENGINE WANTS TO INTERACT WITH AN OBJECT OF THE OTHER ENGINE, IE NEEDS TO MARSHAL OBJECT AND PASS IT BACK TO THE REQUESTING ENGINE

- SO ANY COMMUNICATION BETWEEN JSCRIPT9 AND JSCRIPT ENGINE NEEDS GO THROUGH IE USING `IDispatch` INTERFACE
  1. JSCRIPT USE `IDispatch` IF OF THE HOST TO REQUEST AN OBJECT IN JSCRIPT9 SPACE
  2. HOST USE `IDispatch` IF OF JSCRIPT9 TO MARSHAL A REFERENCE OF THE OBJECT
  3. HOST UNMARSHAL REFERENCE AND RETURNS IT TO JSCRIPT



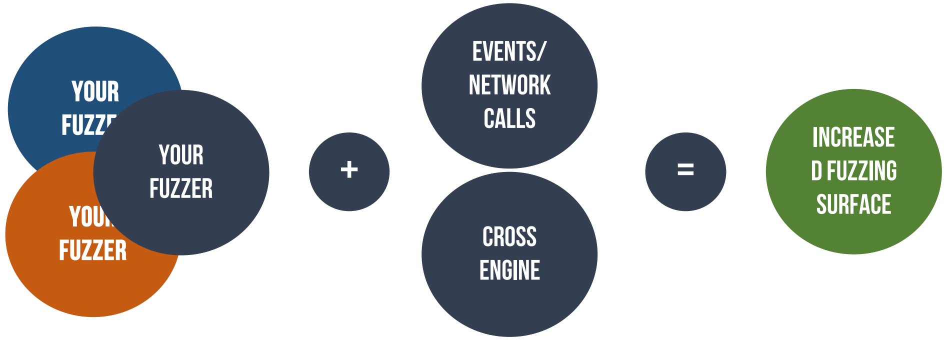
THREAT MODEL

- ANY ENGINE HAS NO KNOWLEDGE OF THE STATUS OF OBJECTS CREATED IN OTHER ENGINES CONTEXTS
- OBJECTS COULD BE DELETED ON THE OTHER ENGINE OR THE WHOLE OTHER ENGINE CONTEXT COULD HAS BEEN DELETED
- IT IS A HOST RESPONSIBILITY TO MAINTAIN CONSISTENCY AMONG OBJECTS AND OBJECTS REFERENCES IN DIFFERENT SCRIPTING CONTEXTS
- TO TRIGGER MEMORY CORRUPTION THE STRATEGY IS TO USE A CLASSIC DOM FUZZING APPROACH BUT ALL CRAWLING-TWEAKING AND MUTATING OPERATIONS ARE PERFORMED CROSS-ENGINE


PARENT WINDOW
JSCRIPT9.DLL
OPEN
USE SEVERAL NAMESPACES
ALSO ON ELEMENTS ON COLLECTIONS
CREATE RANDOM ELEMENTS
TWEAK ELEMENTS ATTRIBUTE S
COLLECT ELEMENT REFERENCE
PERFORM RANDOM MUTATIONS ON DOM
TRIGGER GC
CRAWL DOM REFERENCE
CHILD WINDOW (OFIFRAME)
JSCRIPT.DLL
USE SEVERAL NAMESPACES
ALSO ON ELEMENTS ON COLLECTIONS
CREATE RANDOM ELEMENTS
TWEAK ELEMENTS ATTRIBUTE S
COLLECT ELEMENT REFERENCE
PERFORM RANDOM MUTATIONS ON DOM
TRIGGER GC
CRAWL DOM REFERENCE

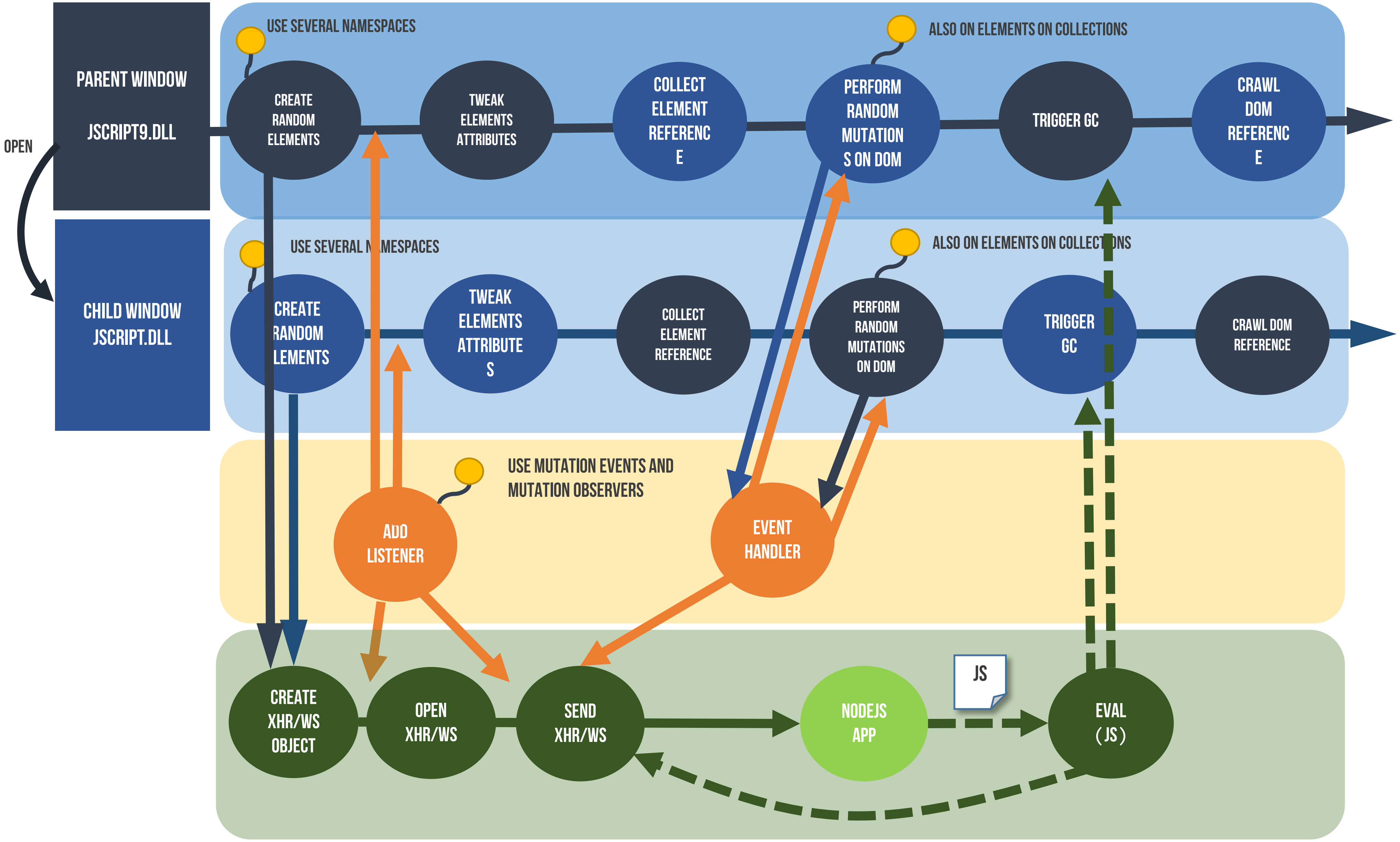
INTRODUCING FILEJA

- A PROTOTYPE COMBINING A TRADITIONAL DOM FUZZING APPROACH WITH TIME EVENTS AND CROSS ENGINES FUZZING
- WRITTEN IN JAVASCRIPT & NODEJS
- TESTED ON GRINDER FRAMEWORK BUT TOTALLY AGNOSTIC FROM FUZZING FRAMEWORK
- COMPLETELY GENERAL TECHNIQUE: NOT BOUND TO SPECIFIC APIs, BUT A PLUG-IN APPROACH TO ANY EXISTING FUZZER



- APPLIED ON NDUJA-LIKE FUZZER
- TWO MONTHS OF FULL TIME TESTING ON MY PC WITH A COUPLE OF WIN7 VMs MAINLY ON IE11 AND CHROME
- 4 EXPLOITABLE BUGS FOUND IN THE FIRST WEEK, 5 MORE IN THE FOLLOWING MONTHS

THE MASTERPLAN #2



SOME RESULTS #1

```
Registers:
    EAX = 0x45454545 -
    EBX = 0x0303FCC0 - RW-
    ECX = 0x68EB08B3 - R-X - jscript9!NativeCodeGenerator::CheckCodeGen
    EDX = 0x019AD684 - RW-
    ESI = 0x00000003 -
    EDI = 0x0AC2F89C - RW-
    EBP = 0x0AC2F720 - RW-
    ESP = 0x0AC2F6D4 - RW-
    EIP = 0x45454545 -

Call Stack:
    0x68EAD364 - jscript9!Js::JavascriptFunction::CallRootFunction
    0x68EAD2B6 - jscript9!ScriptSite::CallRootFunction
    0x68EAD23D - jscript9!ScriptSite::Execute
    0x68EAECC1 - jscript9!ScriptEngineBase::ExecuteInternal<0>
    0x68EAEBFD - jscript9!ScriptEngineBase::Execute
    0x659DE34E - mshtml!CMutationObserver::PerformMicrotaskCheckpoint
    0x659DE284 - mshtml!CObserverManager::InvokeObserversForCheckpoint
    0x65AA45DE - mshtml!GlobalWndOnMethodCall
    0x65487C5E - mshtml!GlobalWndProc
    0x753CC4E7 - user32!InternalCallWinProc
    0x753CC5E7 - user32!UserCallWinProcCheckWow
    0x753CCC19 - user32!DispatchMessageWorker
    0x753CCC70 - user32!DispatchMessageW
    0x65DD8DDC - mshtml!ModelessThreadProc
```

IE 11
TYPE: D.E.P. VIOLATION
EXPLOITABLE: YES

SOME RESULTS #2

IE 11
TYPE: HEAP CORRUPTION
EXPLOITABLE: PROBABLY

```
Registers:
    EAX = 0x02F7AE34 - RW-
    EBX = 0x00000000 -
    ECX = 0x771B179F - R-X - ntdll!vDbgPrintExWithPrefixInternal
    EDX = 0x02F7ABD1 - RW-
    ESI = 0x00620000 - RW-
    EDI = 0x00645478 - RW-
    EBP = 0x02F7AE9C - RW-
    ESP = 0x02F7AE24 - RW-
    EIP = 0x77253873 - R-X - ntdll!RtlReportCriticalFailure

Call Stack:
    0x772547A3 - ntdll!RtlpReportHeapFailure
    0x77254883 - ntdll!RtlpLogHeapFailure
    0x77219D8A - ntdll!RtlpCoalesceFreeBlocks
    0x771E6287 - ntdll!RtlpFreeHeap
    0x771E65A6 - ntdll!RtlFreeHeap
    0x761FC3C4 - kernel32!HeapFree
    0x020E0034 -
    0x03DA9539 -
    0x03DA3167 -
    0x673FCEAB - jscript9!Js::JavascriptFunction::CallFunction<1>
    0x674B46F2 - jscript9!Js::InterpreterStackFrame::Process
    0x67552226 - jscript9!Js::InterpreterStackFrame::OP_TryCatch
    0x674B4712 - jscript9!Js::InterpreterStackFrame::Process
    0x67400AA3 - jscript9!Js::InterpreterStackFrame::InterpreterThunk<1>
```

SOME RESULTS #3

IE 11
TYPE: R.A. VIOLATION
EXPLOITABLE: YES

```
Registers:
    EAX = 0x024C4400 -
    EBX = 0x00000000 -
    ECX = 0x047C87F8 - RW-
    EDX = 0x02C40000 -
    ESI = 0x00000000 -
    EDI = 0x08A2BDA0 - RW-
    EBP = 0x08315D50 - RW-
    ESP = 0x08315D44 - RW-
    EIP = 0x62ACBE7D - R-X - mshtml!CWindow::SetTimeoutWithPaintController

Code:
    0x62ACBE7D - mov eax, [edx]
    0x62ACBE7F - push edx
    0x62ACBE80 - call dword ptr [eax+8]
    0x62ACBE83 - jmp 62a7f579h
    0x62ACBE88 - mov eax, [ecx]
    0x62ACBE8A - push ecx
    0x62ACBE8B - call dword ptr [eax+8]
    0x62ACBE8E - jmp 62a7f58ah
Call Stack:
    0x62A7F6FD - mshtml!CWindow::SetTimeoutHelper
    0x62A7F914 - mshtml!CWindow::SetTimeoutFromScript
    0x62A7F9A6 - mshtml!CFastDOM::CWindow::Trampoline_setTimeout
```

SOME RESULTS #4

IE 11
TYPE: R.A. VIOLATION
EXPLOITABLE: YES

```
Registers:
    EAX = 0x00004444 -
    EBX = 0x0086EAB8 - RW-
    ECX = 0x009320BC - RW-
    EDX = 0x032D1A01 - RW-
    ESI = 0x00000001 -
    EDI = 0x0020003A -
    EBP = 0x02EBC3F8 - RW-
    ESP = 0x02EBC3E8 - RW-
    EIP = 0x6A362B1F - R-X - mshtml!CScriptData::AsyncFireOnError

Code:
    0x6A362B1F - call dword ptr [eax]
    0x6A362B21 - mov esi, eax
    0x6A362B23 - lea ecx, [ebp-8]
    0x6A362B26 - push 0
    0x6A362B28 - push esi
    0x6A362B29 - call mshtml!CElement::CLock::CLock
    0x6A362B2E - push -1
    0x6A362B30 - mov ecx, esi

Call Stack:
    0x69EF1A3C - mshtml!GlobalWndOnMethodCall
    0x69ED9A52 - mshtml!GlobalWndProc
```

SOME RESULTS #5

CHROME 35
TYPE: D.E.P. VIOLATION
EXPLOITABLE: YES

```
Registers:
    EAX = 0x00454545 -
    EBX = 0x00000000 -
    ECX = 0x00410000 - R--
    EDX = 0x00000008 -
    ESI = 0x0020F000 -
    EDI = 0x0020F000 -
    EBP = 0x0015E494 - RW-
    ESP = 0x0015E47C - RW-
    EIP = 0x00454545 -

Call Stack:
    0x668907A2 - chrome_child!WTF::DefaultAllocator::backingMalloc<WebCore::SVGTextChunk
*,void>
    0x66CE6896 - chrome_child!WebCore::MutationObserver::enqueueMutationRecord
    0x66CEE2B6 - chrome_child!WebCore::MutationObserverInterestGroup::enqueueMutationRecord
    0x66AF3188 - chrome_child!WebCore::Element::willModifyAttribute
    0x66B8B668 - chrome_child!WebCore::Element::setAttribute
    0x66B8B29A - chrome_child!WebCore::ElementV8Internal::setAttributeMethod
    0x66B8AFFE - chrome_child!WebCore::ElementV8Internal::setAttributeMethodCallback
```

SOME RESULTS #6

CHROME 35
TYPE: D.E.P. VIOLATION
EXPLOITABLE: PROBABLY

```
Registers:
    EAX = 0x00000000 -
    EBX = 0x00000000 -
    ECX = 0x08010000 -
    EDX = 0x00000008 -
    ESI = 0x08001000 -
    EDI = 0x769BC502 - R-X - kernel32!InterlockedExchangeStub
    EBP = 0x0031E098 - RW-
    ESP = 0x0031E080 - RW-
    EIP = 0x00000000 -

Call Stack:
    0x63914050 - chrome_child!WTF::StringImpl::createUninitialized
    0x641B200F - chrome_child!WebCore::StringTraits<WTF::AtomicString>::fromV8String<0>
    0x63936B84 - chrome_child!WebCore::v8StringToWebCoreString<WTF::AtomicString>
    0x63936CED - chrome_child!WebCore::V8StringResource<1>::operator WTF::AtomicString
    0x63A7B279 - chrome_child!WebCore::ElementV8Internal::setAttributeMethod
    0x63A7AFFE - chrome_child!WebCore::ElementV8Internal::setAttributeMethodCallback
    0x638A4D32 - chrome_child!v8::internal::FunctionCallbackArguments::Call
    0x638A4B76 - chrome_child!v8::internal::HandleApiCallHelper<0>
    0x638A49BB - chrome_child!v8::internal::Builtin_HandleApiCall
```

SOME RESULTS #7

```
Caught a Write Access Violation in process 5896 at 2014-04-12 17:51:14 with a crash hash of
D17449B5.C819B416

Registers:
    EAX = 0x0016F17C - RW-
    EBX = 0xFFFCE3CE -
    ECX = 0x0016F17C - RW-
    EDX = 0x00000004 -
    ESI = 0x00000000 -
    EDI = 0x00C7B000 -
    EBP = 0x0016F15C - RW-
    ESP = 0x0016F14C - RW-
    EIP = 0x6386228C - R-X

Code:
    0x6386228C - mov [edi+4], ebx
    0x6386228F - call 638622ech
    0x63862294 - mov ecx, [ebp+8]
    0x63862297 - mov [edi], ebx
    0x63862299 - mov [ecx], edi
    0x6386229B - add esp, 4
    0x6386229E - pop esi
    0x6386229F - pop ebx
```

CHROME 35
TYPE: W.A. VIOLATION
EXPLOITABLE: PROBABLY

SOME RESULTS #8

SAFARI 5 / WEBKIT
TYPE: R.A. VIOLATION
EXPLOITABLE: YES

```
Registers:
    EAX = 0xFEEEFEEE -
    EBX = 0x02EAA84C - RW-
    ECX = 0x02EAA750 - RW-
    EDX = 0x02EAA750 - RW-
    ESI = 0x02EAA840 - RW-
    EDI = 0x72D9102D - R-X - pthread_mutex_unlock
    EBP = 0x024AF448 - RW-
    ESP = 0x024AF440 - RW-
    EIP = 0x66ECD873 - R-X - CFHostUnscheduleFromRunLoop

Code:
    0x66ECD873 - mov edx, [eax+1ch]
    0x66ECD876 - call edx
    0x66ECD878 - pop ebp
    0x66ECD879 - ret
    0x66ECD87A - int 3
    0x66ECD87B - int 3
    0x66ECD87C - int 3
    0x66ECD87D - int 3

Call Stack:
    0x66ECA6FE - CFHostUnscheduleFromRunLoop
    0x66ECA676 - CFHostUnscheduleFromRunLoop
    0x66EC8F8F - CFHostUnscheduleFromRunLoop
```

SOME RESULTS #9

IE 11 / CHROME 35

TYPE: R.A. VIOLATION

EXPLOITABLE: NO (T YET)

```
Caught a Read Access Violation in process 5036 at 2014-03-29 15:30:59 with a crash hash of
F3D898F3.D2E8590A

Registers:
    EAX = 0x44444444 -
    EBX = 0x00000000 -
    ECX = 0x44444444 -
    EDX = 0x44444444 -
    ESI = 0x02DA2500 - RW-
    EDI = 0x02DA24FC - RW-
    EBP = 0x0360BBEC - RW-
    ESP = 0x0360BBD8 - RW-
    EIP = 0x7719EF10 - R-X - ntdll!RtlEnterCriticalSection


Call Stack:
    0x699FCD4D - mshtml!CMarkup::AcquirePerfData
    0x69936609 - mshtml!URLRequest::ResourceTimingMark
    0x6993AFC7 - mshtml!URLRequest::OnResponse
    0x6993AF9A - mshtml!URLRequest::URLMONRequestSink::OnResponse
    0x7704B075 - urlmon!CINetHttp::QueryStatusOnResponseDefault
    0x77015824 - urlmon!CINetHttp::QueryStatusOnResponse
    0x77015740 - urlmon!CINet::QueryInfoOnResponse
```

FUTURE WORK

- THERE IS NO FINAL VERSION OF FILEJA, THE PROTOTYPE IS JUST AIMED TO SHOW A COUPLE OF CONCEPTS THAT PROVED TO BE EXTREMELY EFFECTIVE IN CRASHING BROWSERS

- PLUG THEM INTO YOUR OWN FUZZER, AND LET IT RUN!

- NOT LIKELY TO CONTINUE THE WORK BY MYSELF, THE FUZZER PROTOTYPE WILL BE PUBLICLY RELEASED

- LEVERAGING ON NETWORK STACK IMPLEMENTATION, A LOT OF TESTING NEEDS TO BE DONE ON NON-WINDOWS PLATFORMS AND MOBILE DEVICES

- THE CROSS- ENGINE APPROACH ON IE CAN BE EXTENDED ALSO TO VBSCRIPT

THANKYOU

谢谢